# OpenStreetMap Data Case Study

**Long Wan, June 1st 2016**

## Map data resource

I chose Nashville, TN, United States as my study resource since I am living and studying here. I am of great interest in looking into this area and finding something interesting. I downloaded relevant XML data from https://mapzen.com/data/metro-extracts/

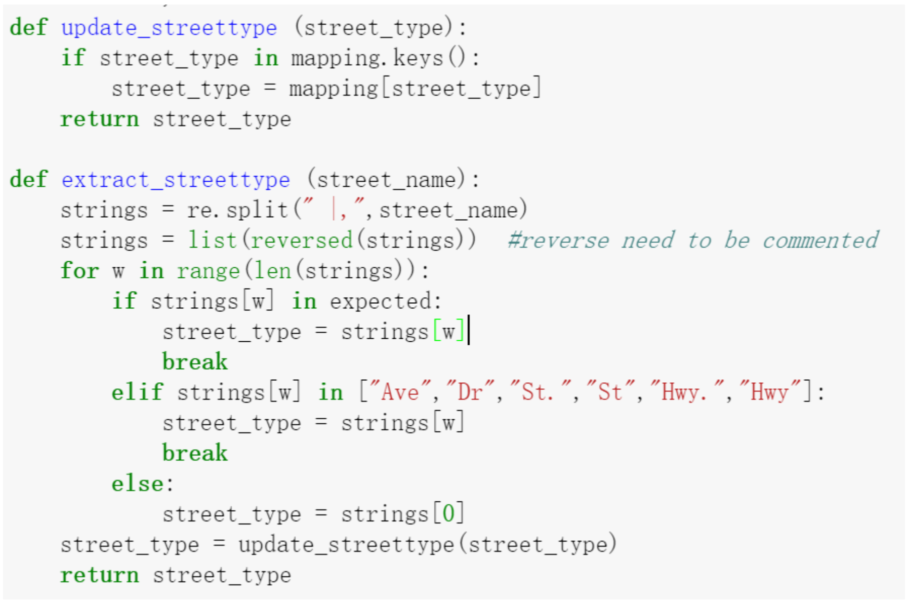
## Problems and cleaning data

By observing the data, I found something problematic regarding to the data format and tried to make them tidy.

- Street formats are not uniform. Some are abbreviated while some are full, such as Franklin Rd vs Bell Road.
- Post code formats are not uniform. Besides regular 5-digit postcode, there are some followed by extra 4-digit number, like *37203-1122*, and one like *TN 37203*, and two obvious errors with only alphabet *TN* as their post number.
- Some values in tags include colons, making the information quite complex and less readable.

### Street formats

Major problem for street name is overabbreviation. My goal is to change road type into regular states, for example, change pky into parkway, rd into road. It means that abbreviation like S would not change to south. In order to extract the street type and fix them, I created a regular name list which contains the full street type name and a mapping list which aimed at switching overabbreviated type names into full names. Then I iterated over each street name and extract street type and transferred those abbreviated names into full names. If the street type is in neither regular list nor mapping list, the last word would be selected as street type.



```
def update_streettype (street_type):
    if street_type in mapping.keys():
        street_type = mapping[street_type]
    return street_type

def extract_streettype (street_name):
    strings = re.split(" |,",street_name)
    strings = list(reversed(strings))  #reverse need to be commented
    for w in range(len(strings)):
        if strings[w] in expected:
            street_type = strings[w]
            break
        elif strings[w] in ["Ave","Dr","St.","St","Hwy.","Hwy"]:
            street_type = strings[w]
            break
        else:
            street_type = strings[0]
    street_type = update_streettype(street_type)
    return street_type
```

The results show that there are still 5 street names in confusion due to incomplete information, like “1305”,”TN 100”,”7th West”. I left them out.

**Post code formats**

I only accept the regular 5-digit number as the correct post code, so I switched all of them into 5-digit number. The two errors with only alphabet TN as post codes are replaced by spaces.

```
import re
pc_type2 = re.compile(r'\d\d\d\d\d')
pc_type3 = re.compile(r'\D\D\ \d\d\d\d\d\Z')
def update_postcode(code):
    if pc_type2.match(code):
        codes = re.search(r'\d\d\d\d\d',code)
        new_code = codes.group()
    elif pc_type3.match(code):
        codes = re.search(r'\d\d\d\d\d',code,re.M|re.I)
        new_code = codes.group()
    else:
        new_code = ""
    return new_code
```

**Value with colons**

I created a variable called type. If the value in one tag is like “addr:street”, It would be separated by the first colon appeared and the type would be “addr” while the value would be “value”. If there is no colon in the value, the type would be “regular”. The LOWER_COLON format was set like this,

LOWER_COLON = re.compile(r'^([a-z]|_)*:([a-z]|_|[0-9]|[A-Z])*$')

As words after the colon could be numbers, uppercase, lowercase and “_” in this case. Below is the part of code to make it, as the entire code is too long to show.

```
if LOWER_COLON.match(child.attrib["k"]):
    if ele.attrib["id"] not in id_list1:
        id_list1.append(ele.attrib["id"])
        f = 0
        node_tag={}
        node_tag["type"]=child.attrib["k"].split(":",1)[0]
        node_tag["key"]=child.attrib["k"].split(":",1)[1]
        node_tag["id"]=ele.attrib["id"]
        node_tag["value"]=child.attrib["v"]
        node_tag["uid"] = f
        node_tags.append(copy.copy(node_tag))
    else:
        node_tag = {}
        f = f + 1
        node_tag["type"]=child.attrib["k"].split(":",1)[0]
        node_tag["key"]=child.attrib["k"].split(":",1)[1]
        node_tag["id"]=ele.attrib["id"]
        node_tag["value"]=child.attrib["v"]
        node_tag["uid"] = f
        node_tags.append(copy.copy(node_tag))
```

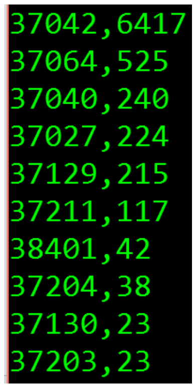
## Features and analysis

### Post code number

First I would like to see places having the greatest number of postcode tags. I used the SQL command to query postcodes with top 10 greatest number.

```
QUERY = "SELECT updated_code, count(*) as num\
         FROM postcode\
         GROUP BY updated_code\
         ORDER BY num DESC\
         LIMIT 10"
```

The results are showed like below,

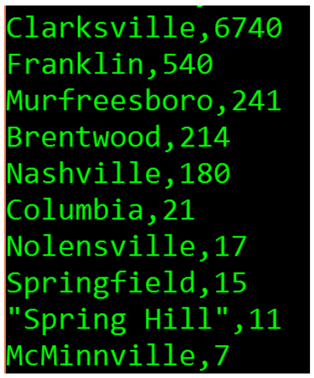


```
37042,6417
37064,525
37040,240
37027,224
37129,215
37211,117
38401,42
37204,38
37130,23
37203,23
```

As we can see, the postcode 37042 has the greatest number of records, which is far more than others. However, 37042 does not lie in Nashville. It is in Clarksville, which is over 40 miles away. Then I check other postcodes and found that only 4 of them lie in traditional great Nashville area. To double check them, I sorted and aggregated samples by county.

```
QUERY = "SELECT tags.value, count(*) as num\
        FROM \
        (SELECT (*)\
        FROM nodes_tags\
        UNION ALL\
        SELECT * FROM way_tags) AS tags\
        WHERE tags.key LIKE "%city"\
        GROUP BY tags.value\
        ORDER BY num DESC\
        LIMIT 10"
```

Here is the result,



```
Clarksville,6740
Franklin,540
Murfreesboro,241
Brentwood,214
Nashville,180
Columbia,21
Nolensville,17
Springfield,15
"Spring Hill",11
McMinnville,7
```

37042 and 37064 represent exactly Clarksville and Franklin, respectively. So instead of saying that this is the data from Nashville, I would rather say that it is the data regarding Clarksville.

Next I want to figure out the reason why 37042 has the abnormally greatest count. I checked the users who built these tags and order them based on the number each of them built.

```
QUERY = "SELECT user, count(*) as num\
        FROM pcinfo\
        WHERE updated_code = '37042'\
        GROUP BY user\
        ORDER BY num DESC"
```

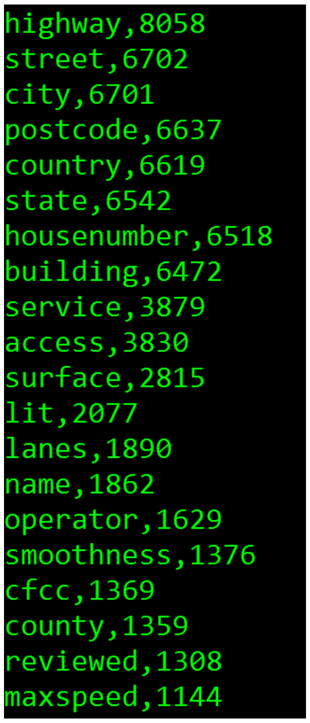
Result is as below,

```
0         Shawn Noble\r  6408
1                 Med\r     2
2              morray\r     2
3              Aury88\r     1
4         RoadGeek_MD99\r   1
5   SomeoneElse_Revert\r    1
6              adjuva\r     1
7            neuhausr\r     1
```

See that 6408 out of overall 6417 tags concerning to 37042 were created by Shawn Noble. It was due to his hard work! I was so curious to his work so I decided to have a look at those tags he created. I would like to know what kind of info and how many of each kind he created on the map data.

```
QUERY = "SELECT key,COUNT(*) as NUM\
        FROM\
        (SELECT id, user FROM nodes\
        WHERE user ='Shawn Noble'\
        UNION ALL\
        SELECT id, user FROM way\
        WHERE user = 'Shawn Noble') AS a\
        LEFT JOIN\
        (SELECT id, key FROM nodes_tags\
        UNION ALL\
        SELECT id, key FROM way_tags) AS b\
        ON a.id = b.id\
        GROUP BY key\
        ORDER BY NUM DESC\
        LIMIT 20"
```

Here is the result,

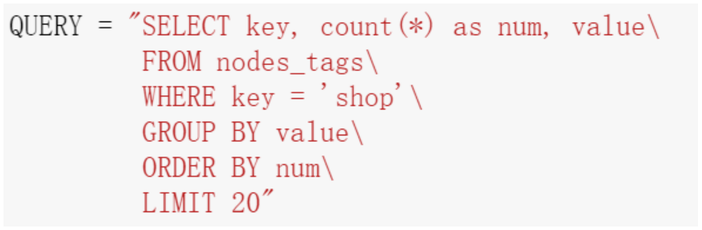


```
highway,8058
street,6702
city,6701
postcode,6637
country,6619
state,6542
housenumber,6518
building,6472
service,3879
access,3830
surface,2815
lit,2077
lanes,1890
name,1862
operator,1629
smoothness,1376
cfcc,1369
county,1359
reviewed,1308
maxspeed,1144
```

See that Shawn created highway information the most. Besides, he created a great amount of address information, like street, housenumber, etc and I believe that these information could be detailed enough for local usage.

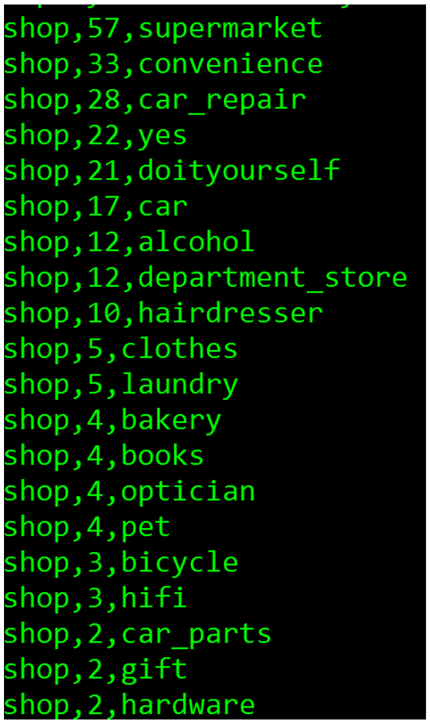
**Shops**

I would like to know the kinds of shops our area have. So I queried the database,



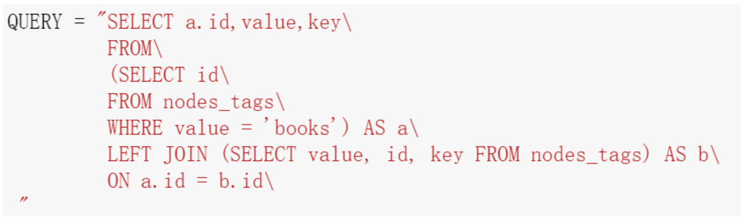
```
QUERY = "SELECT key, count(*) as num, value\
         FROM nodes_tags\
         WHERE key = 'shop'\
         GROUP BY value\
         ORDER BY num\
         LIMIT 20"
```

The results are like this,



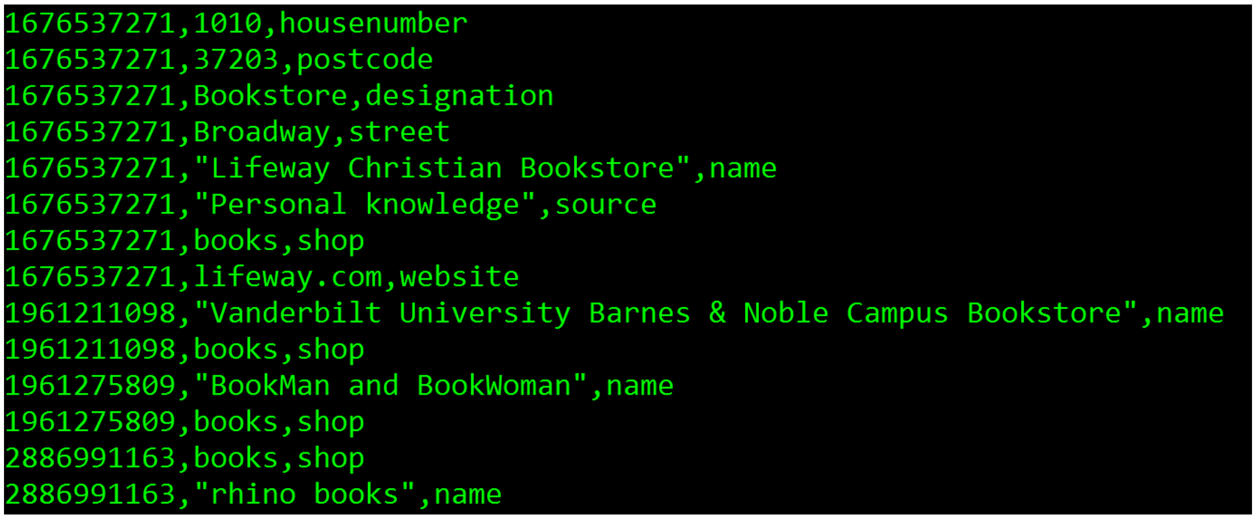
```
shop,57,supermarket
shop,33,convenience
shop,28,car_repair
shop,22,yes
shop,21,doityourself
shop,17,car
shop,12,alcohol
shop,12,department_store
shop,10,hairdresser
shop,5,clothes
shop,5,laundry
shop,4,bakery
shop,4,books
shop,4,optician
shop,4,pet
shop,3,bicycle
shop,3,hifi
shop,2,car_parts
shop,2,gift
shop,2,hardware
```

It seems that Nashville area has a lot of supermarkets and convenience stores. Since I am a student, I care about where I can purchase good books. I know that there a barnes&noble near the Vanderbilt University area, so I would like to know if there are other book stores near my living area.



```
QUERY = "SELECT a.id,value,key\
         FROM\
         (SELECT id\
         FROM nodes_tags\
         WHERE value = 'books') AS a\
         LEFT JOIN (SELECT value, id, key FROM nodes_tags) AS b\
         ON a.id = b.id\
 "
```

The results are like below,



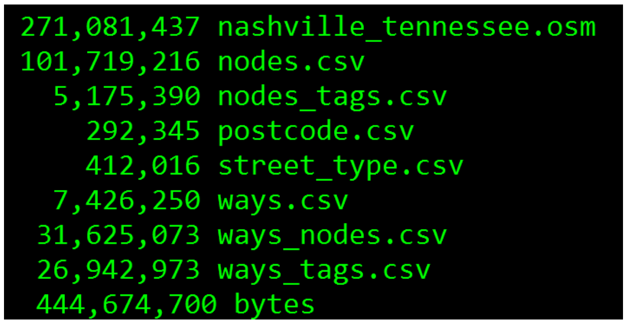
```
1676537271,1010,housenumber
1676537271,37203,postcode
1676537271,Bookstore,designation
1676537271,Broadway,street
1676537271,"Lifeway Christian Bookstore",name
1676537271,"Personal knowledge",source
1676537271,books,shop
1676537271,lifeway.com,website
1961211098,"Vanderbilt University Barnes & Noble Campus Bookstore",name
1961211098,books,shop
1961275809,"BookMan and BookWoman",name
1961275809,books,shop
2886991163,books,shop
2886991163,"rhino books",name
```

As expected, Barnes&Noble is on the list. In addition, I surprisingly found that there is one book store called Lifeway Christian Bookstore locating at 1010 Broadway, very close to my apartment! I went through their website showed on the list and I decided to visit it next time. Unfortunately, addresses are missing for other two book stores.

## Data overview

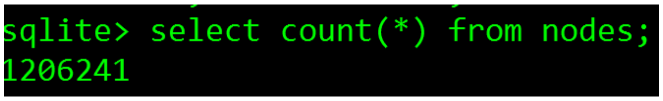
**Full size**

nashville_tennessee.osm 258M



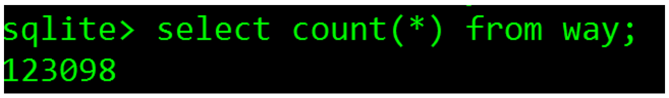
```
271,081,437 nashville_tennessee.osm
101,719,216 nodes.csv
  5,175,390 nodes_tags.csv
    292,345 postcode.csv
    412,016 street_type.csv
  7,426,250 ways.csv
 31,625,073 ways_nodes.csv
 26,942,973 ways_tags.csv
 444,674,700 bytes
```

**Number of nodes**



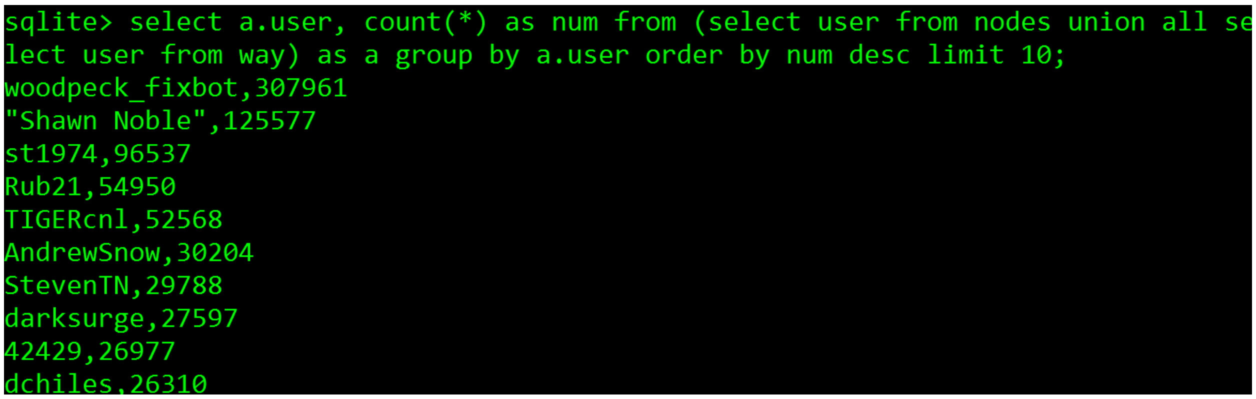
```
sqlite> select count(*) from nodes;
1206241
```

**Number of ways**



```
sqlite> select count(*) from way;
123098
```

**Number of unique users top 10**



```
sqlite> select a.user, count(*) as num from (select user from nodes union all se
lect user from way) as a group by a.user order by num desc limit 10;
woodpeck_fixbot,307961
"Shawn Noble",125577
st1974,96537
Rub21,54950
TIGERcnl,52568
AndrewSnow,30204
StevenTN,29788
darksurge,27597
42429,26977
dchiles,26310
```

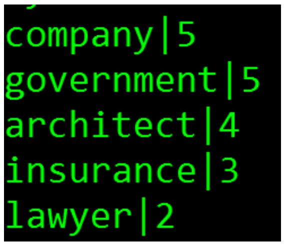
## Suggestions

As I can see from analysis above, the biggest problem of this data is that most of them reflects Clarksville but not real Nashville. The person, Shawn Noble, made a great contribution to the completeness of Clarksville. However, compared to it, a large number of data are still not complete which make them hard to be used by users. For example, 3 out of 4 book stores have no address. So they should encourage more people to help complete these information and make it more balanced and accurate.

## Additional exploration

### Top 10 popular office type

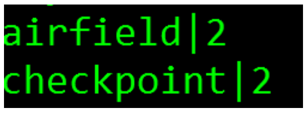
```
QUERY = "SELECT value, count(*) as num\
         FROM node_tags\
         WHERE key = 'office'\
         GROUP BY value \
         ORDER BY num DESC\
         LIMIT 5"
```



```
company|5
government|5
architect|4
insurance|3
lawyer|2
```

### Military facilities

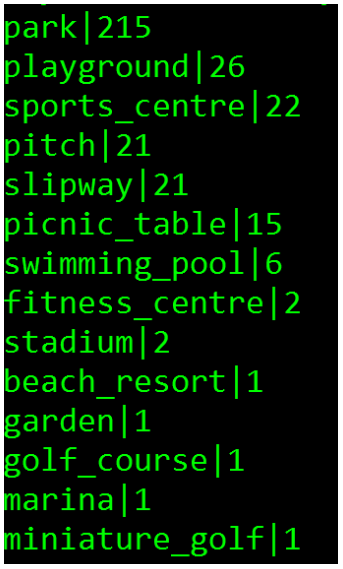
```
QUERY = "SELECT value, count(*) as num\
         FROM node_tags\
         WHERE key = 'military'\
         GROUP BY value"
```



```
airfield|2
checkpoint|2
```

### Leisure types

```
QUERY = "SELECT value, count(*) as num\
         FROM node_tags\
         WHERE key = 'leisure'\
         GROUP BY value\
         ORDER BY num DESC"
```



```
park|215
playground|26
sports_centre|22
pitch|21
slipway|21
picnic_table|15
swimming_pool|6
fitness_centre|2
stadium|2
beach_resort|1
garden|1
golf_course|1
marina|1
miniature_golf|1
```

## Conclusion

The osm data regarding Nashville area is not complete. Most data locate heavily at Clarksville instead of Nashville. A large number of address concerning to shops should be updated. Meanwhile, data is not tidy, so that I spent a large amount of time on cleaning data and setting standardized format repeatedly. More users need to be encouraged to be involved into completing the data.